Sorbonne Université 2021 - Master d'informatique
COMPLEX - Complexité, algorithmes randomisés et approchés

Couverture de graphe

Almehdi KRISNI , Alessia LOI
Groupe 3

Dans ce projet, nous avons choisi d'adopter le langage Python pour implémenter les méthodes demandées.

Notation utilisée dans ce rapport

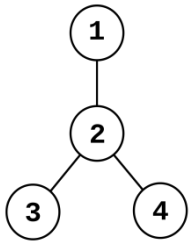
G : graphe non orienté. $G = (V, E)$
V : ensemble de sommets de G
E : ensemble des arêtes de G
n : nombre de sommets dans un graphe G. $n > 0$
m : nombre d'arêtes dans un graphe G
p : probabilité de présence des arêtes dans la génération de graphes aléatoires. $p \in]\,0,\,1\,[$
M : couplage de G, i.e. ensemble d'arêtes $e \in E$ n'ayant pas d'extrémité en commun
C : couverture de G, i.e. ensemble de sommets $s \in V$ tel que toute arête $e \in E$ a au moins une de ses extrémités dans V

2 GRAPHES

Dans notre conception, un graphe est représenté par un dictionnaire $Dict = \{ k : val \}$ dont :

- chaque nœud du graphe est une clé k
- pour chaque clé k, la valeur val est une liste de sommets adjacents au sommet de la clé.

Exemple. Soit un graphe $G = (V, E)$, avec $V = \{ 1, 2, 3, 4 \}$, $E = \{ (1,2), (2,3), (2,4) \}$
Le dictionnaire représentant le graphe G sera : $G = \{ 1: [2], 2: [1,3,4], 3: [2], 4: [2] \}$



Nous nous servons de la librairie Python « networkx » apte à la création et manipulation de graphes, mais nous ne l'utilisons que pour l'affichage graphique des graphes.

3 MÉTHODES APPROCHÉES

3.1 On rappelle que l'algorithme algo-couplage est 2-approché.
Montrer que l'algorithme glouton n'est pas optimal.
En déduire que l'algorithme glouton n'est pas r-approché, pour un certain r à préciser.

Le fonctionnement de l'algorithme glouton est d'ajouter à chaque itération le sommet de degré maximal du graphe dans la couverture. Cela a donc pour effet de supprimer le plus grand nombre d'arêtes possible à chaque itération et de réduire le plus rapidement le nombre d'arêtes du graphe.

On en déduit que la complexité de l'algorithme glouton est en **$O(n^2)$**.

L'algorithme cherche donc à supprimer le plus d'arêtes à chaque itération de la boucle while, ce qui serait efficace dans un graphe dont les sommets ont un grand nombre de voisins. Dans le cas où l'algorithme ferait face à un graphe comportant 2n sommets, étant tous 2 à 2 voisins, le graphe devra donc supprimer les n arêtes du graphe une par une.

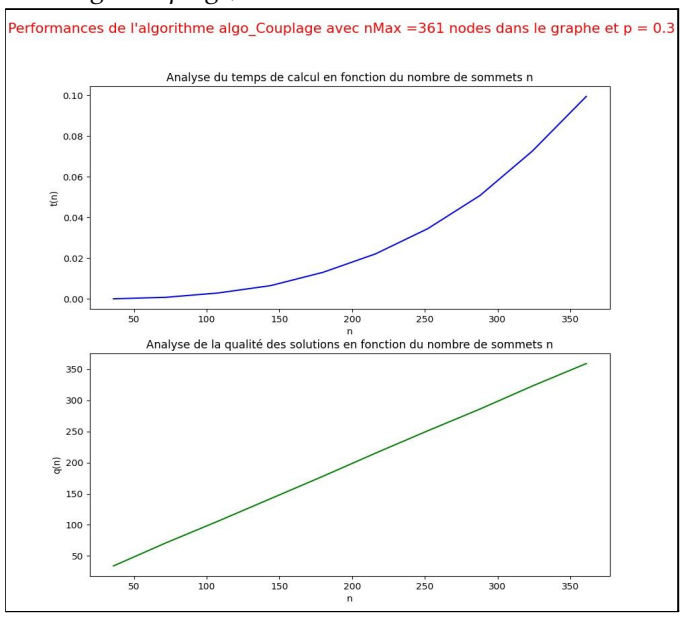
Dans ce cas, on aurait une complexité linéaire de **O(m)** par rapport aux arêtes et donc en **$O(n^2)$** par rapport aux sommets.

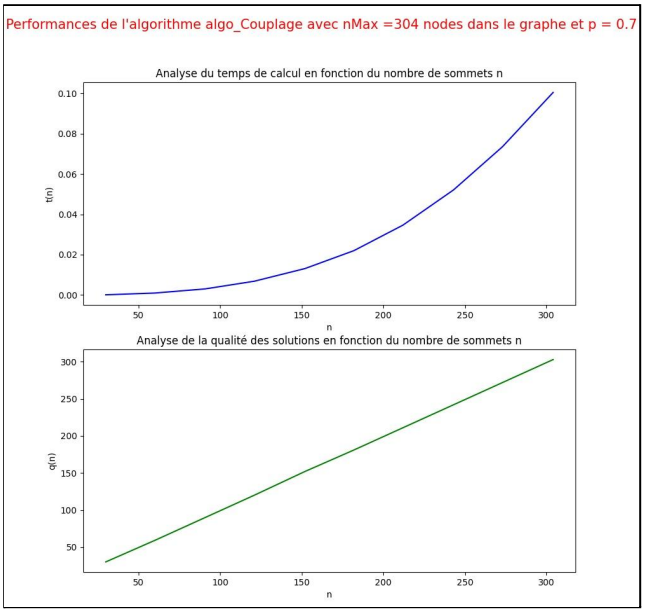
On en conclut que l'algorithme glouton ne renvoie pas toujours une solution optimale puisqu'il est approché et que la taille de sa solution sert de majorant à la solution optimale.

3.2 Comparaison des méthodes algo-couplage et algo-glouton en fonction de n et p du point de vue de leur temps de calcul et de la qualité des solutions retournées.

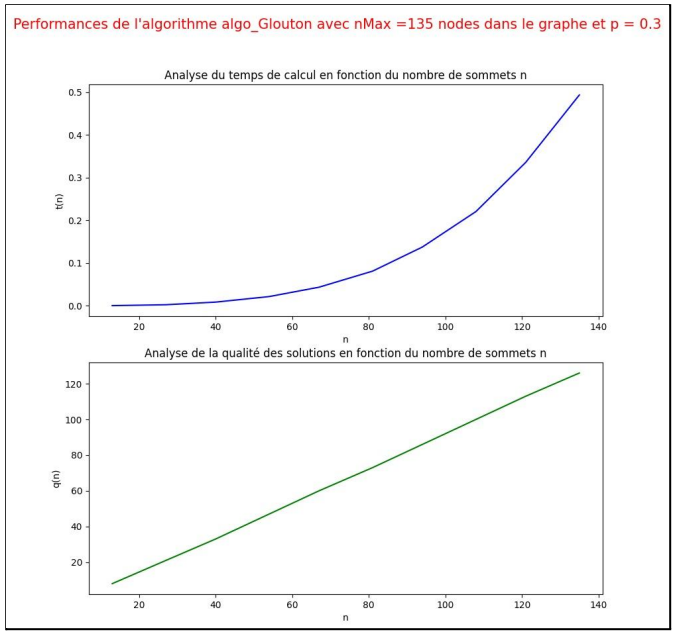
Afin de comparer les deux algorithmes, on décide de les comparer pour un temps maximal d'exécution de **0.1** seconde (allant donc déterminer la valeur de n) et avec $\boldsymbol{p}$ **= { 0.3 , 0.7 }**.

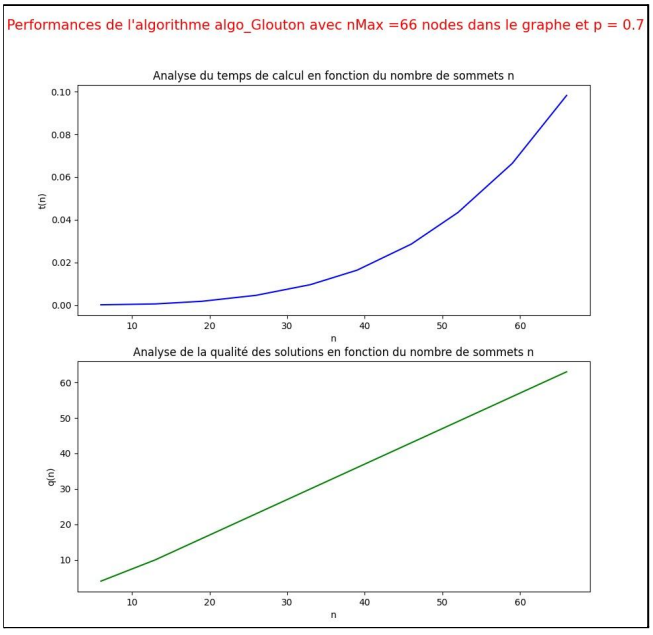
Pour *algo-couplage*, on obtient les résultats suivants :





Pour *algo-glouton*, on obtient les résultats suivants :



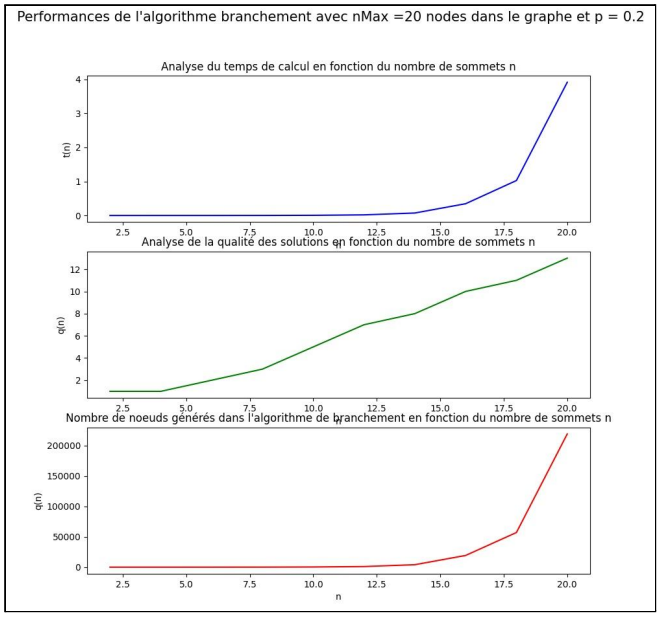


4 SÉPARATION ET ÉVALUATION

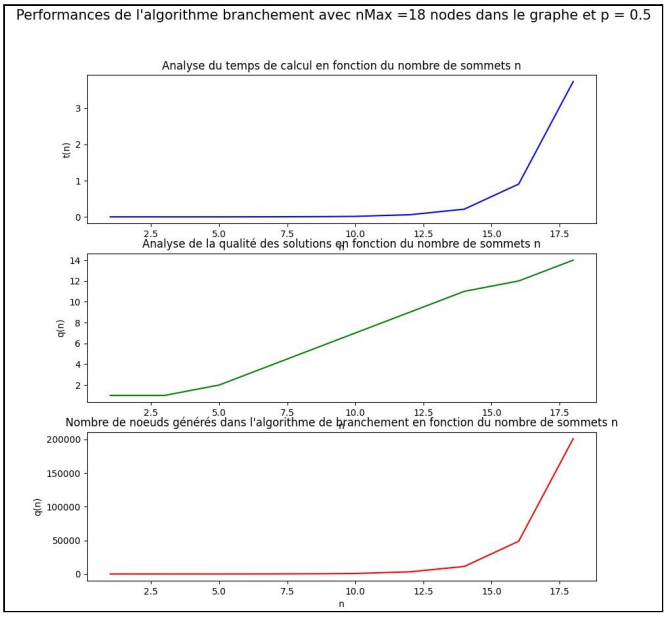
4.1 Branchement

4.1.2 Temps de calcul de la méthode « branchement(G, randomSelection=False) » en fonction de n. Préciser le nombre de nœuds générés.

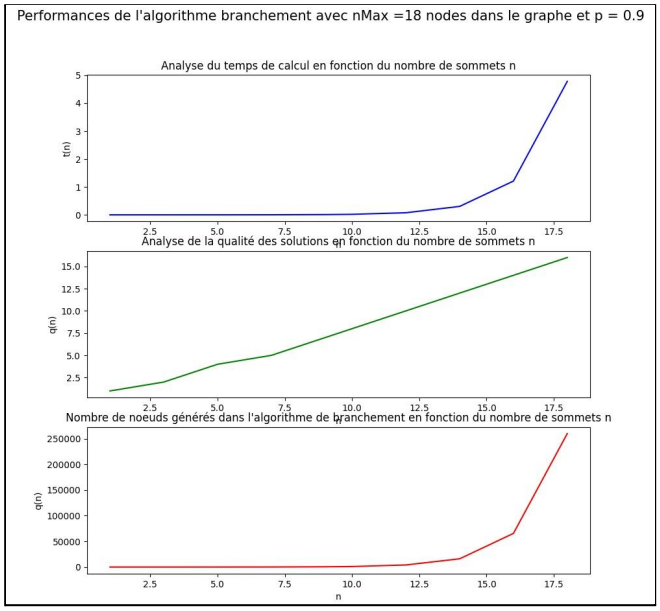
Pour $p = 0.2$, on obtient les résultats suivants :



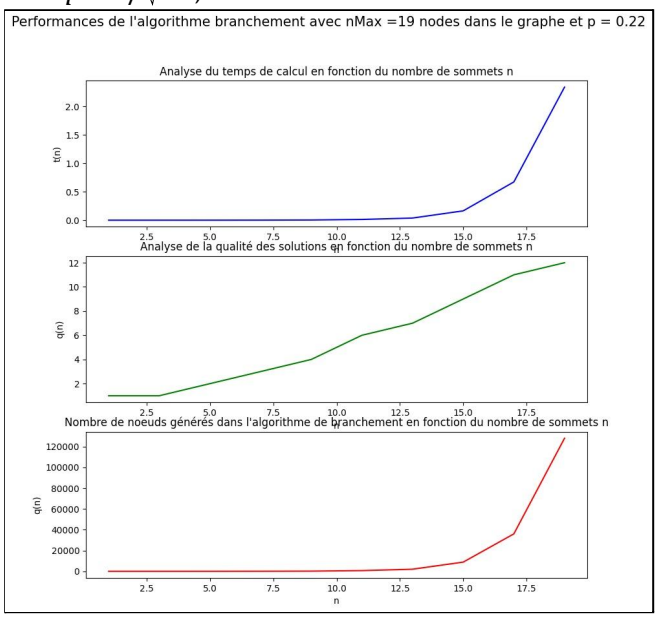
Pour $p = 0.5$, on obtient les résultats suivants :



Pour $p = 0.9$, on obtient les résultats suivants :



Pour $p = 1/\sqrt{n}$, on obtient les résultats suivants :



4.2 Ajout de bornes

4.2.1	Montrer la validité de chacune des bornes b_1 , b_2, b_3. Pour b_3, on pourra s'interroger sur le nombre maximal d'arêtes d'un graphe dont C est une couverture.

Nous avons désormais pour objectif de démontrer la validité des bornes afin de pouvoir élaguer ou non des nœuds dans notre branchement.

Soit un *G* un graphe, *M* un couplage de *G* et *C* une couverture de ***G***, avec *m* le nombre d'arêtes de *G*, *n* le nombre de sommets de *G* et Δ le degré maximal des sommets de *G*.

Preuve de b_1 = partie supérieure (m / Δ) :

Pour k sommets de *G*, le nombre maximal d'arêtes que l'on peut couvrir est de k.Δ puisque le degré maximal est au plus de Δ.

Notons *u* la valeur de la partie supérieure de (m / Δ). Alors pour *u* sommets, on peut couvrir au plus *u*.Δ arêtes, soit au plus m arêtes.

Pour couvrir toutes les arêtes de G, il faut donc au moins *u* sommets. Donc, b_1 **est une borne inférieure**

Preuve de b_2 = |M| :

Cette borne est vérifiée grâce à la définition du couplage maximal.

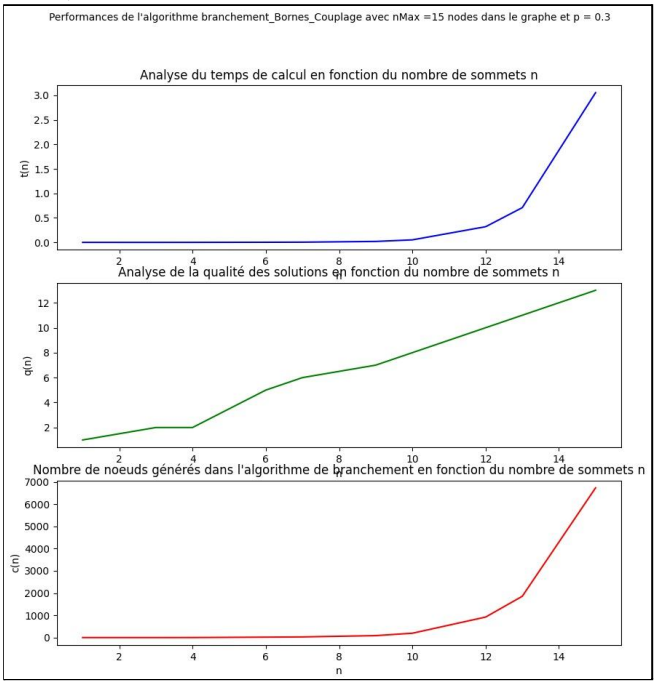
Un couplage maximal d'un graphe est un ensemble d'arêtes n'ayant aucune extrémité en commun et l'ensemble des sommets d'un couplage maximal représente la couverture d'un graphe.

La taille de cette couverture majore donc la taille de la couverture optimale puisque pour chaque arête d'un couplage maximal, au moins une des deux extrémités de l'arête se situe dans la solution optimale (puisque dans le cas contraire, l'arête n'est pas couverte et la solution est invalide) et au plus les deux extrémités sont dans la solution optimale.

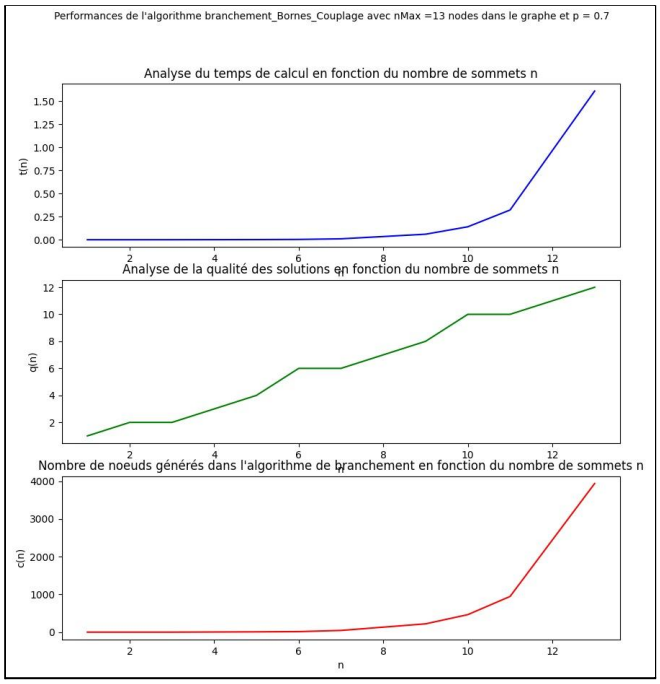
On peut alors confirmer que la cardinalité du couplage M est une borne supérieure de la solution optimale. Donc, b_2**est une borne supérieure.**

4.2.2 Temps de calcul de la méthode « branchementBornesCouplage(G) » en fonction de n. Préciser le nombre de nœuds générés.

Pour $p = 0.3$, on obtient les résultats :



Pour $p = 0.7$, on obtient les résultats :



Grâce aux résultats, on en déduit que la borne supérieure est moins efficace au niveau du temps mais plus efficace au niveau du nombre de sommets explorés.

Cela s'explique du fait que la borne inférieure est évaluée à chaque itération dans la boucle alors que la borne inférieure n'est évaluée qu'une seule fois. L'utilisation de bornes permet donc de trouver un résultat plus rapidement, puisque l'on peut désormais ignorer certains nœuds lors du branchement.

L'élément majeur à noter est la différence du nombre de nœuds générés entre l'algorithme de branchement borné et non-borné. La différence est exponentielle puisque pour 15 sommets, on a 7000 nœuds créés dans l'algorithme borné tandis qu'on en a environ 30000 nœuds créés.

On en conclut que l'algorithme de branchement borné est plus rapide que sa version non-bornée.

4.3 Amélioration du branchement

4.3.1 Temps de calcul de la méthode « brOptimisationCouplage(G) » en fonction de n. Préciser le nombre de nœuds générés.

Pour p = 0.3 :
- avec n = 10, on a 9 noeuds générés et un temps d'exécution de 0.0210 seconde
- avec n = 20, on a 11 noeuds générés et un temps d'exécution de 0.0400 seconde

Pour p = 0.7 :
- avec n = 10, on a on a 14 noeuds générés et un temps d'exécution de seconde 0.0554 seconde
- avec n = 20, on a on a 28 noeuds générés et un temps d'exécution de 37.0619 secondes

4.3.2 Temps de calcul de la méthode «brOptimiseCouplage_uDegreMax(G) » en fonction de n. Préciser le nombre de nœuds générés.

Pour p = 0.3 :
- avec n = 10, on a 10 noeuds générés et un temps d'exécution de 0.022 seconde
- avec n = 20, on a 44 noeuds générés et un temps d'exécution de 15.31667 secondes

Pour p = 0.7 :
- avec n = 10, on a 8 noeuds générés et un temps d'exécution de 0.0405 seconde
- avec n = 20, on a 18 noeuds générés et un temps d'exécution de 31.02042 secondes

4.3.3 Dans un graphe G, si un sommet u est de degré 1, expliquer pourquoi il existe toujours une couverture optimale qui ne contient pas u.

Soit un graphe G contenant l'arête {u ,v} et u un sommet de degré 1 de G. On peut alors en déduire que toute couverture optimale de G contient au moins l'un des deux sommets de l'arête. Le sommet u ne couvre que l'arête {u, v} tandis que le sommet v couvre au minimum l'arête {u, v}.

On suppose alors une couverture optimale C de G contenant le sommet u. On peut alors remplacer u par v dans C et être assuré que C reste une couverture optimale de G puisque la taille de C n'a pas changé et toutes les arêtes de G sont toujours couvertes.

4.4 Qualité des algorithmes approchés

4.4.1 Évaluer expérimentalement le rapport d'approximation des algorithmes de couplage et glouton en fonction de n.
Donner le pire rapport d'approximation obtenu lors des expérimentations, pour chacun des deux algorithmes.